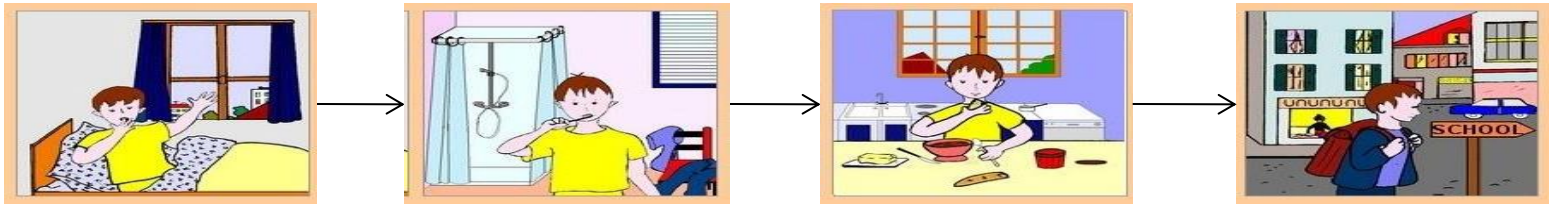
CSE101-Part-1(control structure)

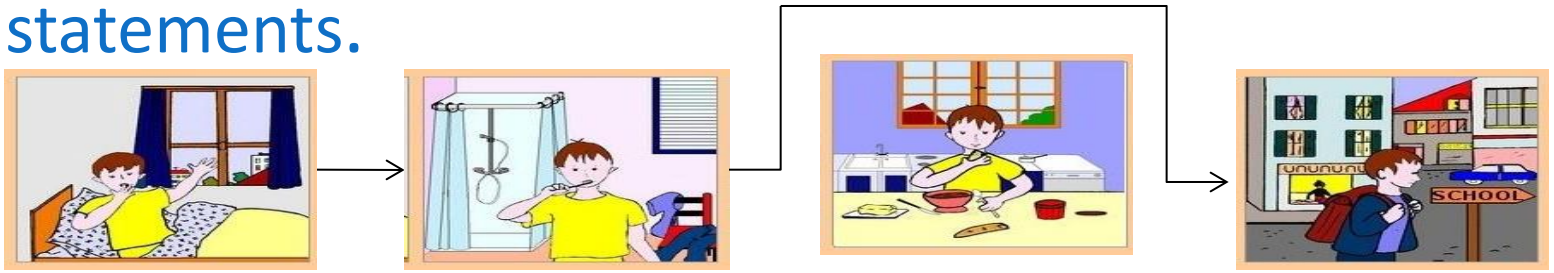
- Control structures(Decision control statements/ or Condition Statements)

Program

- Program is a set of instruction executed one by one.



- Depending upon the circumstances sometimes it is desirable to alter the sequence of execution of statements.



1. Wake up;
2. Get ready;
3. If you have enough time, then eat breakfast;
4. Go to school.

Control Statements

- The C language programs until now follows a sequential form of execution of statements.
- C language provides statements that can alter the flow of a sequence of instructions. These statements are called control statements.
- These statements help to jump from one part of the program to another. The control transfer may be conditional or unconditional.

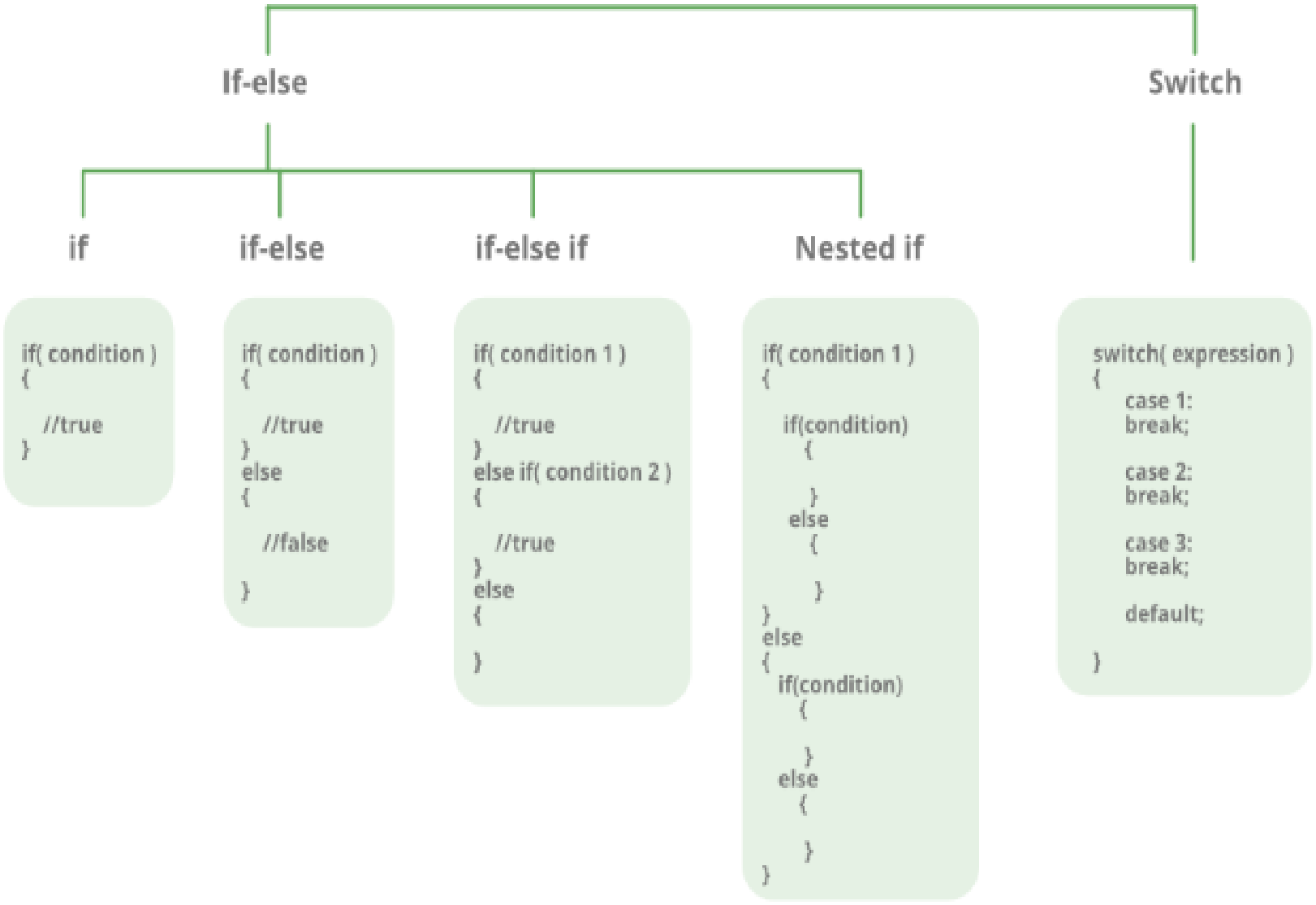
Control Structure

- A control structure refers to the way in which the programmer specifies the order of executing the statements.
- Three control structures
  - Sequence structure
    - Programs are executed sequentially by default.
  - Selection structures(Condition)
    - `if`, `if…else`, if-else-if, Nested-if ,`switch`
  - Repetition structures (iteration)
    - `while`, `do…while`, `for`

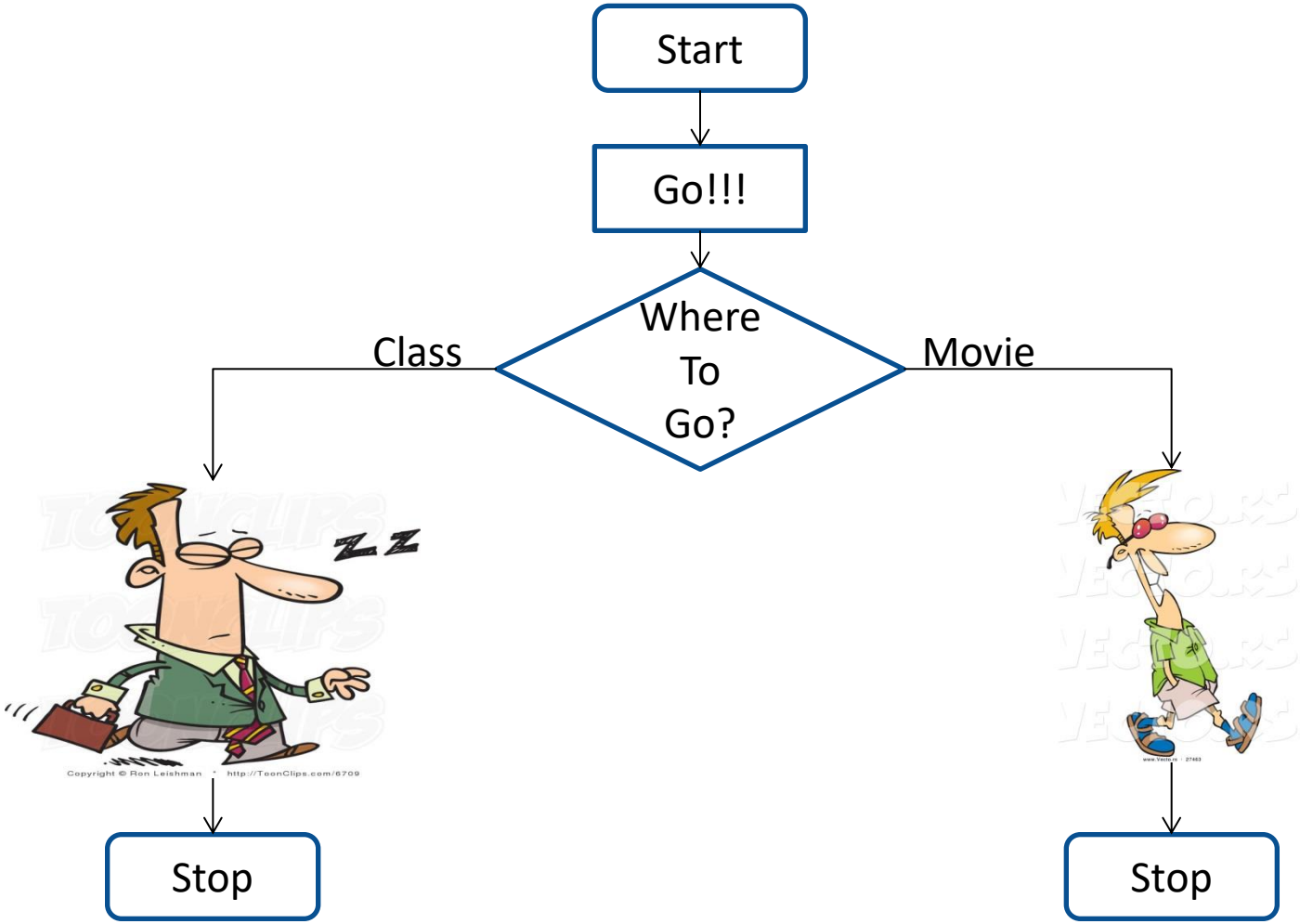
Condition Statements(or Decision control statements or Branching statements)

- The C condition statements or the decision statements, checks the given condition
- Based upon the state of the condition, a sub-block is executed.
- Decision statements are the:
  - *if statement*
  - *if-else statement*
  - *If-else-if statement*
  - *Nested if statement*
  - *switch statement*

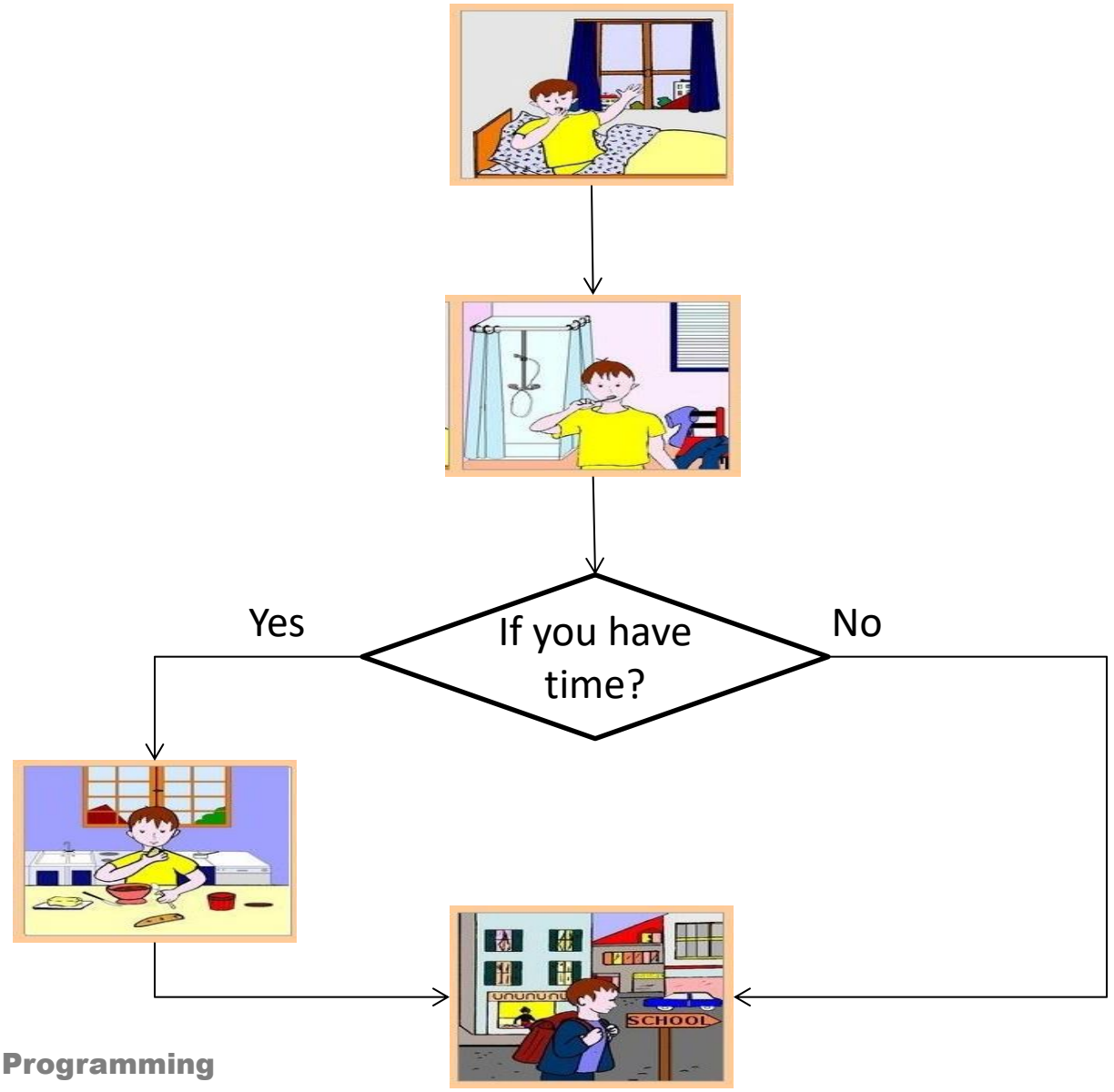
Decision Making



Daily routine



if statement



`if` Statement

- If statement
  - It is decision making statement uses keyword `if`.
  - It allows the computer to evaluate the expression first
    - and then, depending on whether the value is 'true' or 'false', i.e. non zero or zero it transfers the control to a particular statement.

A decision can be made on any expression.
zero - `false`
nonzero - `true`
Example:
`3 < 4` is `true`

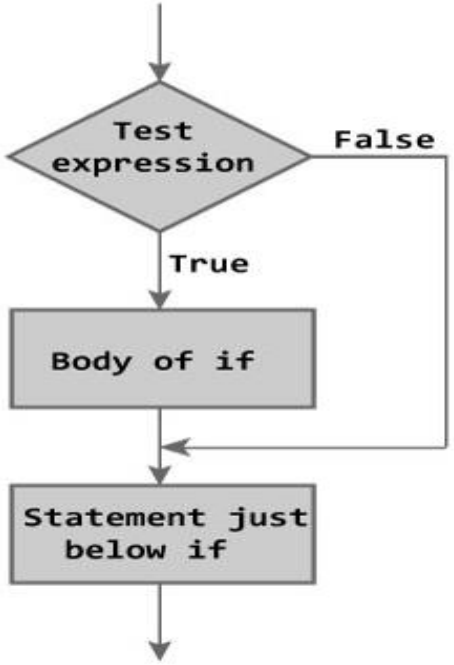
if Statement

Syntax

```
if (expression)
statement;

or

if (expression)
{
  block of statements;
}
```



`if` Statement

- The *if statement* has the following syntax:

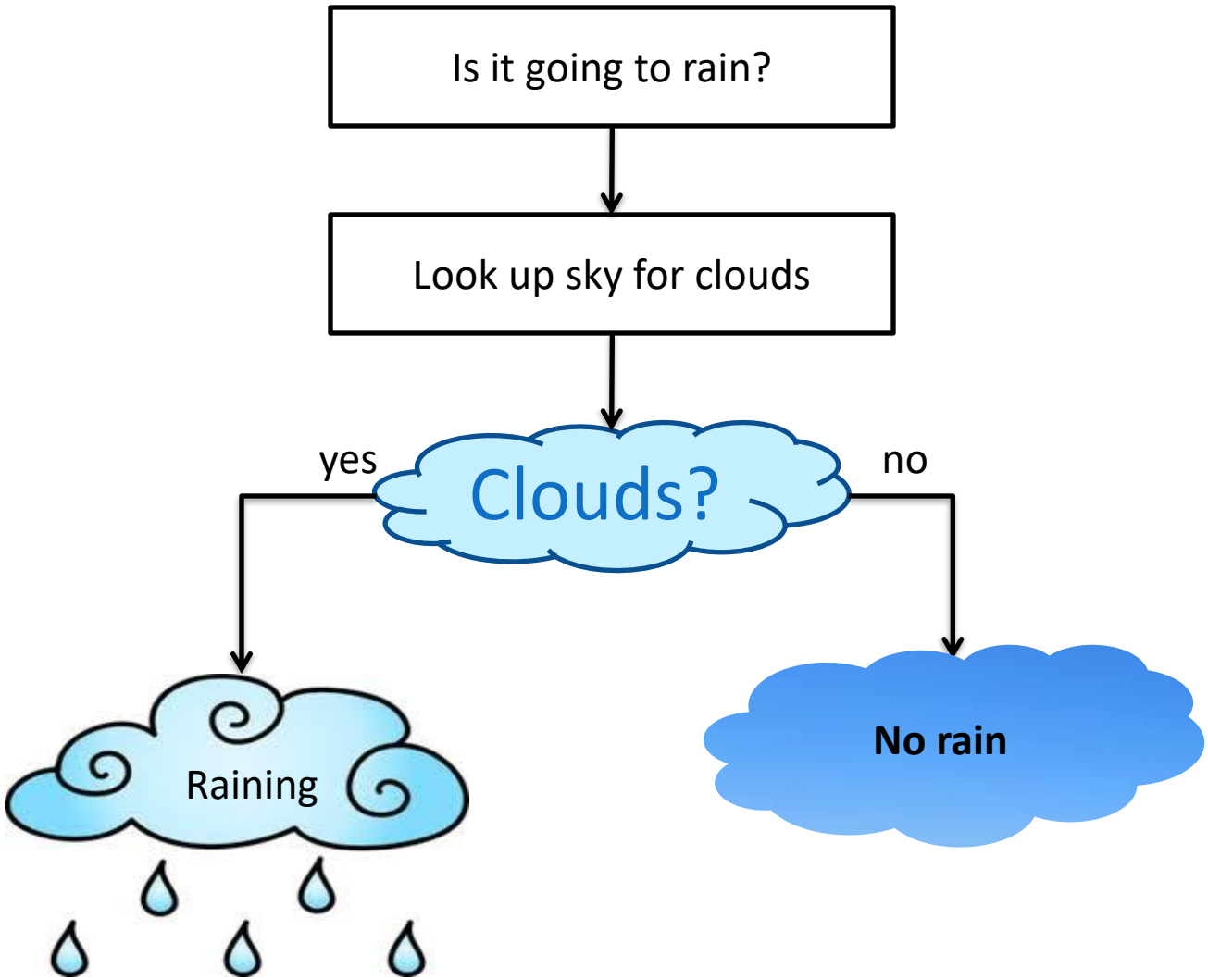
`if` is a C reserved word

The *`condition`* must be a boolean expression. It must Evaluate to either non-zero or zero.

```
if ( condition )/* no semi-colon */
    statement;
```

If the *`condition`* is non-zero, the *`statement`* is executed. If it is zero, the *`statement`* is skipped.

Rain ???

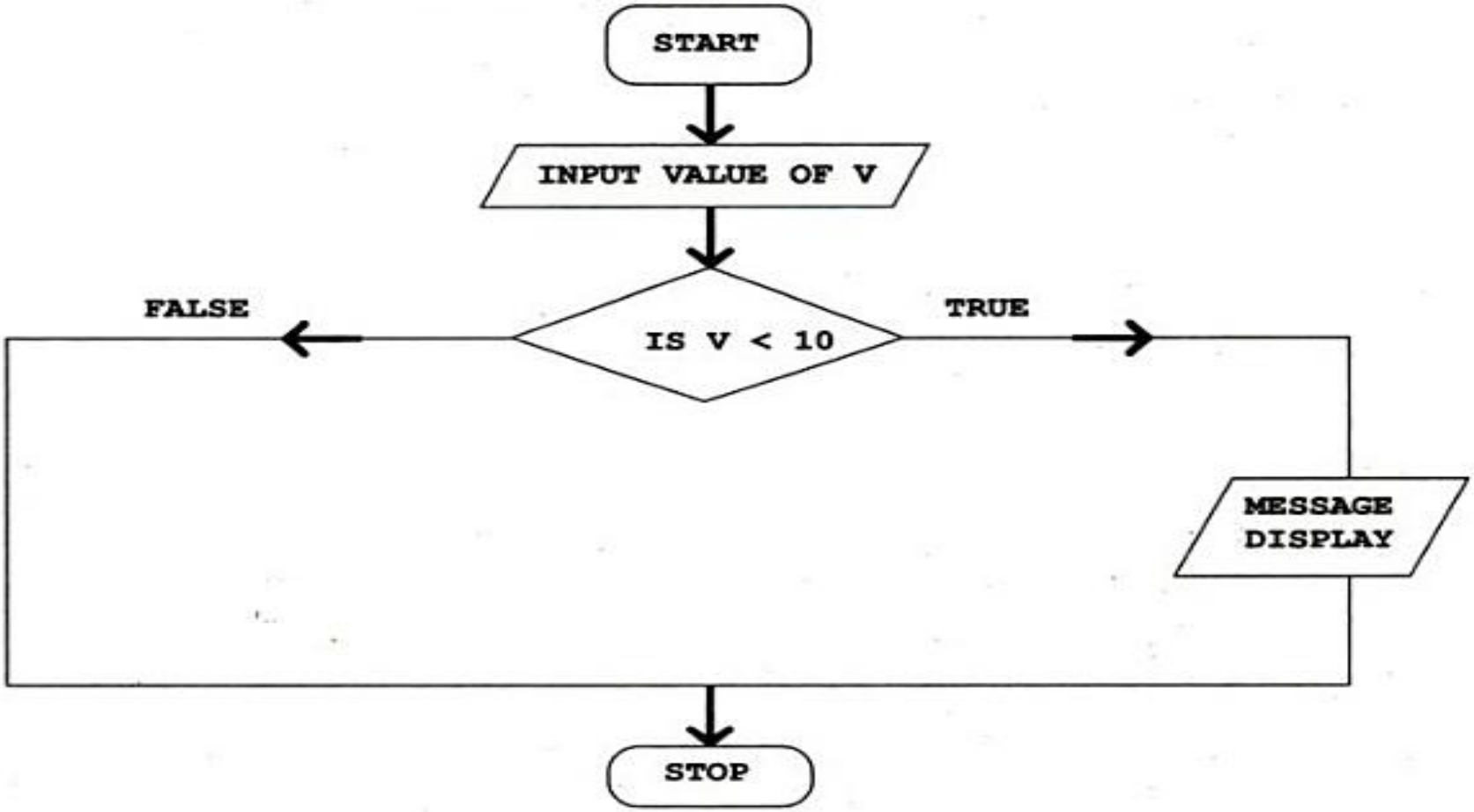

Is it going to rain?
Look up sky for clouds
Clouds?
yes
no
Raining
No rain

```
#include<stdio.h>
int main()
{
 int v;
 printf("Enter the number :");
 scanf("%d", &v);
 if(v<10)
   printf("number is less  than 10");
return 0;
}
```

Program to check whether number is less than 10.

```
Enter the number: 6
Number is less than 10
```

Control Flow



`if..else` statement

If you have time?

Yes

No

Grab something to eat along

`if..else` statement

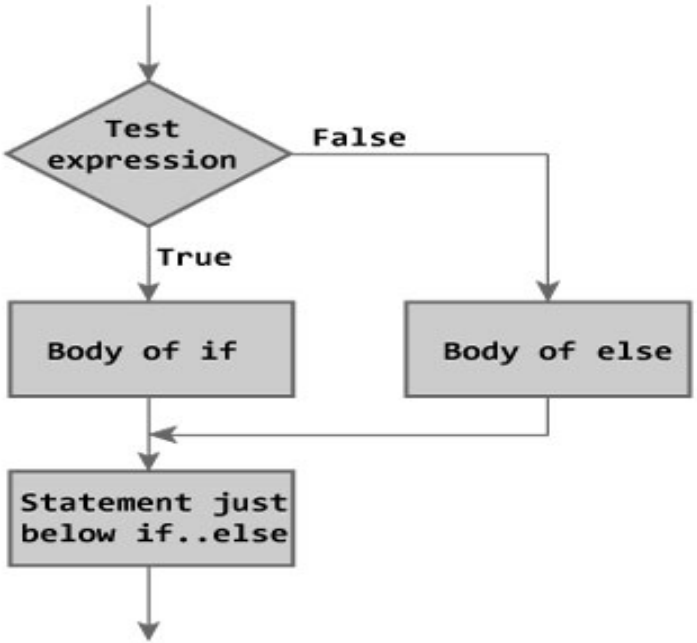
- The `if` statement executes only when the condition following if is true.
- It does nothing when the condition is false.
- The `if..else` statement takes care of the true and false conditions.

`if..else` statement

- `if..else` has two blocks.
- One block is for `if` and it is executed when condition is **non-zero**(true).
- The other block is of `else` and its executed when condition is **zero** (false).

Syntax

```
if (expression)
{
  block of statements;
}
else
{
  block of statements;
}
```



`if..else` statement

- The `else` statement cannot be used without `if`.
- No multiple `else` statements are allowed with one `if`.
- `else` statement has no expression.
- Number of `else` cannot be greater than number of `if`.

```
#include<stdio.h>
int main()
{
 int a;
 printf("Enter the number :");
 scanf("%d", &v);
 if(v<10)
   printf("number is less than 10");
 else
   printf("number is greater than 10");
return 0;
}
```

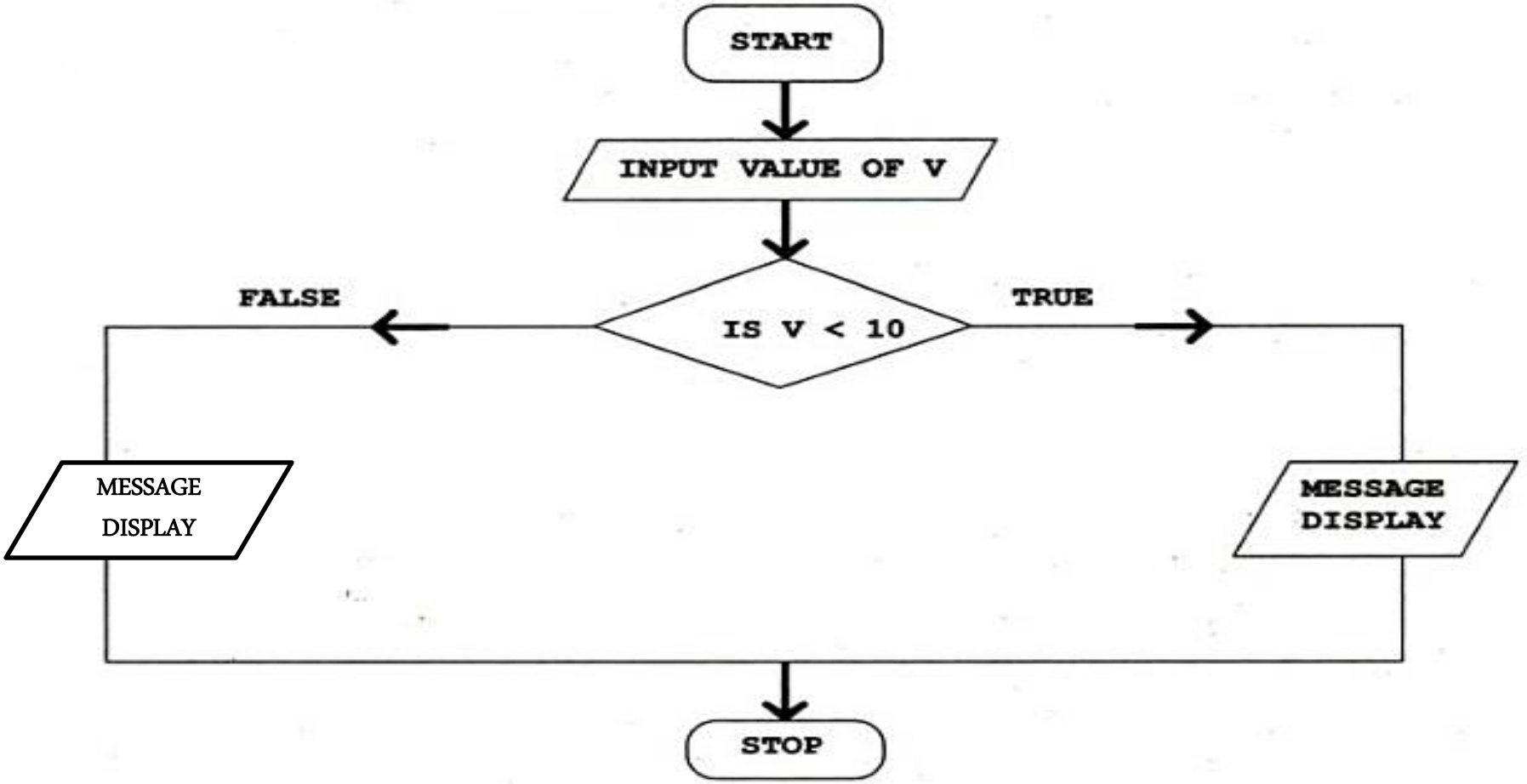
Example : Program to check whether number is less than 10.

```
Enter the number: 7
Number is less than 10
```

or

```
Enter the number: 100
Number is greater than 10
```

Control Flow



Q1

What will be the output of the following C code?

```
#include <stdio.h>
int main()
{
    int x = 5;
    if (x < 1)
        printf("hello");
    if (x == 5)
        printf("hi");
    else
        printf("no");
    return 0;
}
```

A. hi

B. hello

C. no

D. error

Q2

What will be the output of the following C code?

```
#include <stdio.h>
int main()
{
    int x = 0;
    if (x == 0)
        printf("hi");
    else
        printf("how are u");
        printf("hello");
    return 0;
}
```

A. hi

B. how are you

C. hello

D. hihello

Q3

What will be the output of the following C code?

```
#include <stdio.h>
int main()
{
    int x = 5;
    if (x < 1);
        printf("Hello");

}
```

A. Nothing will be printed

B. Compile time error

C. Hello

D. Logical error

Q4

```
#include<stdio.h>
int main()
{
float x=2.3;
if(x==2.3)
{
printf("Hi");
}
else
{
printf("Hello");
}
return 0;
}
```

A. Hi
B. Hello
C. Compile time error
D. None of these

Q5

```
#include<stdio.h>
int main()
{
int x=-1;
if(x)
{
printf("Hi");
}
else
{
printf("Hello");
}
return 0;
}
```

A. Hi
B. Hello
C. Compile time error
D. None of these

Q6

What is the output of this C code?

```
#include <stdio.h>
int main()
{
    float f = 0.1;
    if (f == 0.1)
        printf("True");
    else
        printf("False");
    return 0;
}
```

A. True

B. False

C. Compile time error

D. None of these

If-else-if

- `if-else-if` statement is used when program requires more than one test expression.
- We can check multiple conditions, and what so ever condition is true, that part will work
- Here, a user can decide among multiple options. The C if statements are executed from the top down. As soon as one of the conditions controlling the if is true, the statement associated with that if is executed, and the rest of the C else-if ladder is bypassed. If none of the conditions are true, then the final else statement will be executed.

If-else-if ladder

Syntax

```
if ( condition ) {
    block of statements;
}
else if ( condition ) {
    block of statements;
}
else {
    block of statements;
}
```

```
#include<stdio.h>
int main()
{
 int v;
 printf("Enter the number :");
 scanf("%d", &v);
 if(v<10){
   printf("number is less than 10");
 }
 else if(v<100){
   printf("number is less than 100");
 }
return 0;
}
```

Program to check whether number is less than 10.

```
Enter the number: 1
Number is less than 10
```

or

```
Enter the number: 56
Number is less than 100
```

Program to print grades of students marks.

```
#include<stdio.h>
int main()
{
 float marks;
 scanf("%f", &marks);
 if (marks>90){
      printf("Grade A");
    }
 else  if (marks>80) {
       printf("Grade B");
      }
 else  if(marks>70){
       printf("Grade C");
      }
 else if (marks >60) {
       printf("Grade D");
      }
return 0;
}
```

```
66.70
Grade D
```

or

```
78.00
Grade C
```

Q1

```
#include <stdio.h>
   int main()
   {
      int x = 1;
      if (x > 0)
         printf("inside if\n");
      else if (x > 0)
         printf("inside elseif\n");
   }
```

A. inside if

B. inside elseif

C. inside if
   inside elseif

D. Compile time error

Q2

What will be the output of the following C code?

```
#include <stdio.h>
int main()
{
    int x = 0;
    if (x++)
        printf("true\n");
    else if (x == 1)
        printf("false\n");
}
```

A. true

B. false

C. Compile time error

D. undefined behaviour

Q3

What will be the output of the following C code?

```
#include <stdio.h>
int main()
{
    int x = 0;
    if (x == 0)
        printf("true, ");
    else if (x = 10)
        printf("false, ");
    printf("%d\n", x);
    return 0;
}
```

A. false, 0

B. true, 0

C. true, 10

D. compile time error

Nested if

- A nested if in C is an if statement that is the target of another if statement. Nested if statements means an if statement inside another if statement. C allows us to nested if statements within if statements, i.e, we can place an if statement inside another if statement.

Syntax

```
if (condition1)
{
   // Executes when condition1 is true
   if (condition2)
   {
      // Executes when condition2 is true
   }
}
```

Program example

```
// C program to illustrate nested-if statement
#include <stdio.h>

int main()
{
        int i = 10;
        if (i == 10)
        {
        // First if statement
        if (i < 15)
        printf("i is smaller than 15\n");

// Nested - if statement
// Will only be executed if statement above  is true
        if (i < 12)
        printf("i is smaller than 12 too\n");
        else
        printf("i is greater than 15");
        }
        return 0;
}
```

What will be the output of following code?

```
#include <stdio.h>
   int main()
   {
      int x = 0;
      if (x == 1)
         if (x >= 0)
            printf("true\n");
         else
            printf("false\n");
   }
```

A. true

B. false

C. Depends on the compiler

D. Nothing will be printed

What will be the output of the following C code?

```
#include <stdio.h>
int main()
{
    int x = 0;
    if (x == 1)
        if (x == 0)
            printf("inside if\n");
        else
            printf("inside else if\n");
    else
        printf("inside else\n");
    return 0;
}
```

A.inside if

B.inside else if

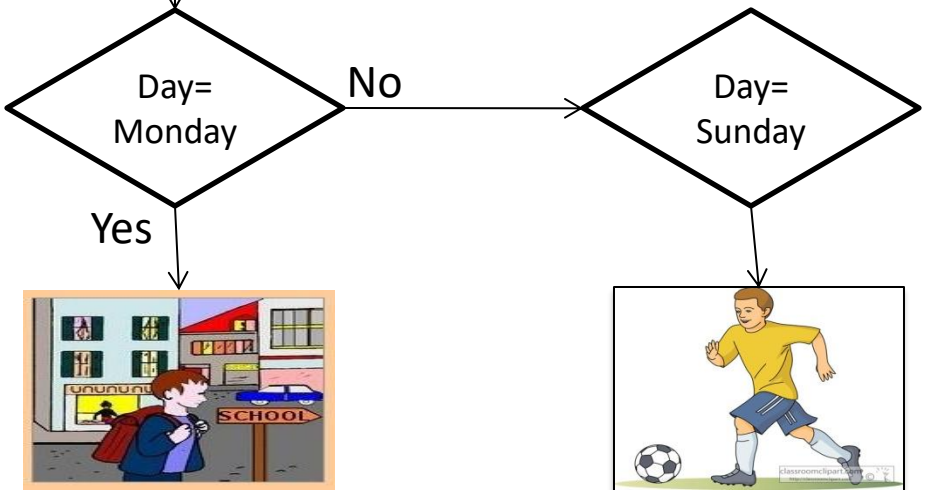
C.inside else

D.Compile time error

`break` statement

- **`break`** is a **keyword**.
- `break` allows the programmer to terminate the loop.
- A `break` statement causes control to transfer to the first statement after the loop or block.
- The `break` statement can be used in nested loops. If we use `break` in the innermost loop then the control of the program is terminated only from the innermost loop.

switch Statement



`switch` Statement

- The control statement that allows to make a decision from the number of choices is called switch.
- Also called switch-case-default.
- The switch statement provides another way to decide which statement to execute next.
- The switch statement evaluates an expression, then attempts to match the result to one of several possible cases.
- Each case contains a value and a list of statements.
- The flow of control transfers to statement associated with the first case value that matches.

switch Statement

Syntax

```
switch (expression)
{
case constant1:
        statements;
        break;
case constant2:
        statements;
        break;
case constant3:
        statements;
        break;
default:
        statements;
}
```

switch and case are reserved words

```
switch ( expression )
{
   case value1 :
      statement-list1
   case value2 :
      statement-list2
   case value3 :
      statement-list3
   case  ...

}
```

If *expression* matches *value2*, control jumps to here

Rules of using `switch case`

1. Case label must be **unique**
2. Case label must end with colon
3. Case label must have **constant expression**
4. Case label must be of integer, character type like case 2, case 1+1, case ‘a’
5. Case label should not be **floating point**
6. Default can be placed anywhere in switch
7. Multiple cases cannot use same expression
8. Nesting of switch is allowed.
9. **Variables** are not allowed in switch case label..

Syntax error in `switch` statement

```
switch(pt){
      case count:
       printf("%d", count);
       break;

      case 2.5:
       printf("A line");
       break;
      case 3 + 7.7:
       printf("A triangle");
      case 3 + 7.7:
       printf("A triangle");
       break;
      case count+5:
       printf("A pentagon");
       break;
}
```

Variable cannot be used as label

Floating point number cannot be used

Floating point number cannot be used and same expression cannot be used

constant expression should be used

Program to show switch statement in geometry

```
#include<stdio.h>
int main()
{
   int pt;
   printf("Enter the number of nodes:");
   scanf("%d", &pt);
   switch(pt){
     case 0:
      printf("\nNo Geometry");
     break;
     case 1:
      printf("\nA point");
     break;
     case 2:
      printf("\nA line");
     break;
     case 3:
      printf("\nA triangle");
     break;
     case 4:
      printf("\nA rectangle");
     break;
     case 5:
      printf("\nA pentagon");
     break;
    default:
     printf("Invalid input");
     break;
    }
return 0;
}
```

```
Enter the number of nodes: 2
A line
```

Q1

```
#include <stdio.h>
  int main()
  {
     double ch;
     printf("enter a value between 1 to 2:");
     scanf("%lf", &ch);
     switch (ch)
     {
       case 1:
         printf("1");
         break;
       case 2:
         printf("2");
         break;
     }
     return 0;
  }
```

A. Compile time error

B. 1

C. 2

D. Nothing will be displayed

Q2

What will be the output of the following C code? (Assuming that we have entered the value 1 in the standard input)

```
#include <stdio.h>
int main()
{
   int ch;
   printf("enter a value between 1 to 2:");
   scanf("%d", &ch);
   switch (ch)
   {
     case 1:
       printf("1 ");
     default:
       printf("2");
   }
   return 0;
}
```

A. 1

B. 2

C. 1 2

D. Compile time error

Q3

What will be the output of the following C code? (Assuming that we have entered the value 1 in the standard input)

```
#include <stdio.h>
int main()
{
   int ch;
   printf("enter a value between 1 to 2:");
   scanf("%d", &ch);
   switch (ch)
   {
     case 1:
       printf("1 ");
       printf("hi");
       break;
     default:
       printf("2\n");
   }
}
```

A. 1 hi

B. 2

C. hi

D. 1

Q4

What will be the output of the following C code?

```
#include <stdio.h>
int main()
{
   int x = 97;
   switch (x)
   {
     case 'a':
       printf("yes ");
       break;
     case 97:
       printf("no");
       break;
   }
}
```

A. yes

B. yes no

C. Duplicate case value error

D. Nothing will be displayed

Q5

What will be the output of the following C code?

```
#include <stdio.h>
int main()
{
   int a = 1;
   switch (a)
   {
     case a:
       printf("Case A ");
     default:
       printf("Default");
   }
  return 0;
}
```

A. Output: Case A

B. Output: Default

C. Output: Case A Default

D. Compile time error